

What Is Claimed Is:

1 1. A method for selectively monitoring store instructions to support
2 transactional execution of a process, comprising:
3 encountering a store instruction during transactional execution of a block
4 of instructions in a program, wherein changes made during the transactional
5 execution are not committed to the architectural state of a processor until the
6 transactional execution successfully completes;
7 determining whether the store instruction is a monitored store instruction
8 or an unmonitored store instruction;
9 if the store instruction is a monitored store instruction,
10 performing a corresponding store operation, and
11 store-marking a cache line associated with the store
12 instruction to facilitate subsequent detection of an interfering data
13 access to the cache line from another process; and
14 if the store instruction is an unmonitored store instruction, performing the
15 corresponding store operation without store-marking the cache line.

1 2. The method of claim 1, wherein prior to executing the program, the
2 method further comprises generating the instructions for the program, wherein
3 generating the instructions involves:
4 determining whether store operations that take place during transactional
5 execution need to be monitored;
6 generating monitored store instructions for store operations that need to be
7 monitored; and
8 generating unmonitored store instructions for store operations that do not
9 need to be monitored.

1 3. The method of claim 2, wherein determining whether a store
2 operation needs to be monitored can involve examining a data structure associated
3 with the store operation to determine whether the data structure is a “protected”
4 data structure for which stores need to be monitored, or an “unprotected” data
5 structure for which stores do not need to be monitored.

1 4. The method of claim 2, wherein determining whether a store
2 operation needs to be monitored can involve determining whether the store
3 operation is directed to a heap, wherein stores from the heap need to be monitored
4 and stores from outside the heap do not need to be monitored.

1 5. The method of claim 2, wherein determining whether a store
2 operation needs to be monitored can involve allowing a programmer to determine
3 if the store operation needs to be monitored.

1 6. The method of claim 1, determining whether the store instruction is
2 a monitored store instruction involves examining an op code of the store
3 instruction.

1 7. The method of claim 1, determining whether the store instruction is
2 a monitored store instruction involves examining an address associated with the
3 store instruction to determine whether the address falls within a range of addresses
4 for which stores are monitored.

1 8. The method of claim 7, wherein examining the address involves
2 comparing the address with one or more boundary registers.

1 9. The method of claim 7, wherein examining the address involves
2 examining a Translation Lookaside Buffer (TLB) entry associated with the
3 address.

1 10. The method of claim 1, wherein if an interfering data access from
2 another process is encountered during transactional execution of the block of
3 instructions, the method further comprises:
4 discarding changes made during the transactional execution; and
5 attempting to re-execute the block of instructions.

1 11. The method of claim 1, wherein if transactional execution of the
2 block of instructions completes without encountering an interfering data access
3 from another process, the method further comprises:
4 committing changes made during the transactional execution to the
5 architectural state of the processor; and
6 resuming normal non-transactional execution of the program past the
7 block of instructions.

1 12. The method of claim 1, wherein an interfering data access can
2 include:
3 a store by another process to a cache line that has been load-marked by the
4 process; and
5 a load or a store by another process to a cache line that has been store-
6 marked by the process.

1 13. The method of claim 1, wherein the cache line is store-marked in
2 the cache level closest to the processor where cache lines are coherent.

1 14. The method of claim 1, wherein a store-marked cache line
2 indicates at least one of the following:
3 loads from other processes to the cache line should be monitored;
4 stores from other processes to the cache line should be monitored; and
5 stores to the cache line should be buffered until the transactional execution
6 completes.

1 15. An apparatus that selectively monitors store instructions to support
2 transactional execution of a process, comprising:
3 an execution mechanism within a processor;
4 wherein the execution mechanism is configured to support transactional
5 execution of a block of instructions in a program, wherein changes made during
6 the transactional execution are not committed to the architectural state of a
7 processor until the transactional execution successfully completes;
8 wherein upon encountering a store instruction during transactional
9 execution, the execution mechanism is configured to,
10 determine whether the store instruction is a monitored store
11 instruction or an unmonitored store instruction,
12 if the store instruction is a monitored store instruction, to
13 perform a corresponding store operation, and to store-mark a cache
14 line associated with the store instruction to facilitate subsequent
15 detection of an interfering data access to the cache line from
16 another process; and

17 if the store instruction is an unmonitored store instruction,
18 to perform the corresponding store operation without store-
19 marking the cache line.

1 16. The apparatus of claim 15, further comprising an instruction
2 generation mechanism configured to:
3 determine whether store operations that take place during transactional
4 execution need to be monitored;
5 generate monitored store instructions for store operations that need to be
6 monitored; and to
7 generate unmonitored store instructions for store operations that do not
8 need to be monitored.

1 17. The apparatus of claim 16, wherein the instruction generation
2 mechanism is configured to determine whether a store operation needs to be
3 monitored by examining a data structure associated with the store operation to
4 determine whether the data structure is a “protected” data structure for which
5 stores need to be monitored, or an “unprotected” data structure for which stores do
6 not need to be monitored.

1 18. The apparatus of claim 16, wherein the instruction generation
2 mechanism is configured to determine whether a store operation needs to be
3 monitored by determining whether the store operation is directed to a heap,
4 wherein stores from the heap need to be monitored and stores from outside the
5 heap do not need to be monitored.

1 19. The apparatus of claim 16, wherein the instruction generation
2 mechanism is configured to determine whether a store operation needs to be
3 monitored by allowing a programmer to determine if the store operation needs to
4 be monitored.

1 20. The apparatus of claim 15, wherein the execution mechanism is
2 configured to determine whether a store operation needs to be monitored by
3 examining an op code of the store instruction.

1 21. The apparatus of claim 15, wherein the execution mechanism is
2 configured to determine whether a store operation needs to be monitored by
3 examining an address associated with the store instruction to determine whether
4 the address falls within a range of addresses for which stores are monitored.

1 22. The apparatus of claim 21, wherein the execution mechanism is
2 configured to examine the address by comparing the address with one or more
3 boundary registers.

1 23. The apparatus of claim 21, wherein the execution mechanism is
2 configured to examine the address by examining a Translation Lookaside Buffer
3 (TLB) entry associated with the address.

1 24. The apparatus of claim 15, wherein if an interfering data access
2 from another process is encountered during transactional execution of the block of
3 instructions, the execution mechanism is configured to:
4 discard changes made during the transactional execution; and to
5 attempt to re-execute the block of instructions.

1 25. The apparatus of claim 15, wherein if transactional execution of
2 the block of instructions completes without encountering an interfering data
3 access from another process, the execution mechanism is configured to:
4 commit changes made during the transactional execution to the
5 architectural state of the processor; and to
6 resume normal non-transactional execution of the program past the block
7 of instructions.

1 26. The apparatus of claim 15, wherein an interfering data access can
2 include:
3 a store by another process to a cache line that has been load-marked by the
4 process; and
5 a load or a store by another process to a cache line that has been store-
6 marked by the process.

1 27. The apparatus of claim 15, wherein the cache line is store-marked
2 in the cache level closest to the processor where cache lines are coherent.

1 28. The apparatus of claim 15, wherein a store-marked cache line
2 indicates at least one of the following:
3 loads from other processes to the cache line should be monitored;
4 stores from other processes to the cache line should be monitored; and
5 stores to the cache line should be buffered until the transactional execution
6 completes.

1 29. A computer system that selectively monitors store instructions to
2 support transactional execution of a process, comprising:
3 a processor;
4 a memory;
5 an execution mechanism within the processor;
6 wherein the execution mechanism is configured to support transactional
7 execution of a block of instructions in a program, wherein changes made during
8 the transactional execution are not committed to the architectural state of a
9 processor until the transactional execution successfully completes;
10 wherein upon encountering a store instruction during transactional
11 execution, the execution mechanism is configured to,
12 determine whether the store instruction is a monitored store
13 instruction or an unmonitored store instruction,
14 if the store instruction is a monitored store instruction, to
15 perform a corresponding store operation, and to store-mark a cache
16 line associated with the store instruction to facilitate subsequent
17 detection of an interfering data access to the cache line from
18 another process; and
19 if the store instruction is an unmonitored store instruction,
20 to perform the corresponding store operation without store-
21 marking the cache line.